A Mobile and Web-Based Approach for Targeted and Proactive Participatory Sensing

Navid Hashemi Tonekaboni(✉), Lakshmish Ramaswamy, and Sakshi Sachdev

Department of Computer Science, University of Georgia, Athens, GA, USA
{navidht, sss11759}@uga.edu, laks@cs.uga.edu

Abstract. Participatory sensing applications have gained popularity due to the increased use of mobile phones with embedded sensors. One of the main issues in participatory sensing applications is the uneven coverage of areas, i.e., some areas might be covered by multiple participants while there is no data for other areas. In this paper, we design mobile and web-based infrastructure to enable domain scientists to effectively acquire crowd-sensed data from specific areas of interest (AOIs) to support the goal of even coverage for data collection. Scientists can mark the AOIs on a web-portal, then volunteers will be proactively informed about the participatory sensing opportunities near their current location. We presented a caching algorithm to increase the performance of our proposed system and studied the performance of the caching algorithm for different real-world scenarios on different mobile phones. We observed that prefetching data improves the performance to some extent; however, it starts to degrade after a certain point depending upon the number of nearby AOIs.

Keywords: Participatory sensing · Mobile caching · Citizen science · Crowdsensing

1 Introduction

Mobile phones have evolved from merely being a medium of audio communication to a means of improved information exchange between individuals or groups using the Internet, along with accessing the GPS, microphones, cameras, accelerometer, and other sensors. Due to these additional features, mobile phones are being widely used by users in their day-to-day lives. The increase of mobile phones with embedded sensors has introduced a paradigm called participatory sensing [1]. The main idea of participatory sensing is empowering citizens for collecting data from ubiquitous, handheld mobile phones [2], and it is used in various domains such as urban planning, public health, and natural resource management [3, 4]. Participatory sensing can also be referred to as crowdsensing, urban sensing, community sensing, people-centric sensing, opportunistic sensing, or citizen sensing [5, 6].

Participatory sensing has many advantages over traditional sensor networks. First, due to the mobility of users, broad areas can be covered [7]. Secondly, more often than not, there is no need to deploy and maintain sensors as they are integrated into mobile phones. Lastly, the availability of software development tools for mobile phones makes application development and deployment relatively easy [7]. Advantages of participatory sensing have led to an increase in mobile sensing applications. Maintaining user's privacy, recruiting and training participants, incentivizing them, dealing with low-quality data, and interpreting the data are some of the challenges in such applications [1].

There are many different challenges in participatory sensing applications. Quality of the crowd-sensed data, the anonymity of users, incentivization mechanisms, and resource consumption are all different factors to be taken into account in designing such applications. This study focuses on collecting data only from the areas where domain scientists are interested in, as an essential step towards efficient resource consumption. The data collected from the same area by different participants waste participants' resources. On the other hand, the resource consumption of mobile applications plays an essential role in keeping or losing users. Therefore, our approach can significantly benefit different participatory sensing applications. Most of the current participatory sensing applications are passive, and there is a little or even no communication between the participants and domain scientists. In passive participatory sensing, participants install the application and submit the geotagged data to the backend server, while they have no idea whether the geo-tagged data is redundant or not. Passive mode of operation causes data disparity; too much data may get collected from specific locations, while the data acquired from many other locations are insufficient.

In this study, we propose a mobile and web-based approach for leveraging domain scientists' areas of interest to address this shortcoming of many participatory sensing applications. A web portal is developed through which researchers can specify the AOIs to enable proactive participatory sensing. We have also designed and implemented a mobile caching algorithm to prefetch the AOIs as participants are collecting data. To analyze the caching performance, different implementations of the algorithm have been tested on three mobile devices under different scenarios. Our analysis shows the caching performance starts to degrade after a certain threshold depending upon the number of nearby AOIs.

2 Background

In this section, we briefly discuss some of the existing participatory sensing applications and also illustrate the system model of the typical applications. Participatory sensing applications can be classified into three areas based on the type of data being collected [1]:

Environment Centric Applications. In this type of applications, sensors collect data from a surrounding environment of participants. For instance, Youdale et al. [11] introduced a participatory sensing application called Haze Watch in which a mobile phone is interfaced with a pollution sensor to measure carbon monoxide, oxygen, and nitrogen dioxide in the air as well as the temperature and wind speed. This information is uploaded to a server along with the time and location of the participant, which is used by environmental scientists and ecologists. Another application called Ikarus [12] collects thermal information of the atmosphere during flights by paraglider pilots, which is used by other pilots to gain the required heights. It uploads this information along with the time and location of pilots to be used for navigation purposes. Noise Tube [13] and Noise Spy [14] are two other applications of this category which are used to monitor noise pollution. Participants record the audio and upload it to the server. These data are used by specialists to understand the relationship between noise exposition and behavioral problems. Another example is Creek Watch [15], an application that is used for water and trash management. In this application, users take pictures of creeks at various locations and upload this information to the server. All the applications mentioned above are designed to monitor and analyze a specific environmental phenomenon.

Urban Centric Applications. These applications focus on infrastructure and urban information. ParkNet [8] is one of such applications that provides information about parking space occupancy to users through an ultrasonic device installed on the cars. As a result, users can find the nearest vacant parking spots. Nericell [9] is another location-based service app which monitors road traffic conditions through smartphones. It uses microphones, accelerometer, GSM radio, and GPS sensors to detect potholes and bumps. These data are reported to the server to be aggregated for annotating maps and allowing users to search for best driving directions.

Community Centric Applications. These are people-centric applications which use sensor devices to collect data about users. Diet Sense [17] is one such application which captures the image of foods that users eat along with the time, date, and location. Participants share these images with community members to compare their eating habits. The primary use case of this application is for diabetic patients and the ones who want to lose weight using the suggestions from other people with the same conditions. MobAsthma [21] is a personalized asthma-monitoring application which lets asthma specialists and allergists explore the relationship between respiratory symptoms and exposure to different air pollutions. It also monitors the person’s asthma condition and remotely alerts the medical staff if the patient experiences an asthma attack. BikeNet [18] is another application that monitors sports exercise of participants by analyzing location information, speed, and burnt calories. It also measures the carbon dioxide on the route taken by the cyclists to find the most suitable routes for cycling. In another application of this category called Live Compare [10], participants take pictures of products’ price and barcode. The app searches the stores in that proximity to display the current price of that product in other stores.

Overview of Typical Participatory Sensing Model. Typical participatory sensing applications have a client- server architecture in which mobile devices act as clients through which the participants collect sensor data and submitted to the application server. In the backend server, the data are stored, analyzed, and made available in various forms to the end users as shown in Fig. 1. Various stakeholders are involved in participatory sensing applications as follows:

Participants/Citizen Scientists. Participants act as information providers. They gather sensor data for participatory sensing application and submit it through Wi-Fi or wireless operators to the infrastructure provider.

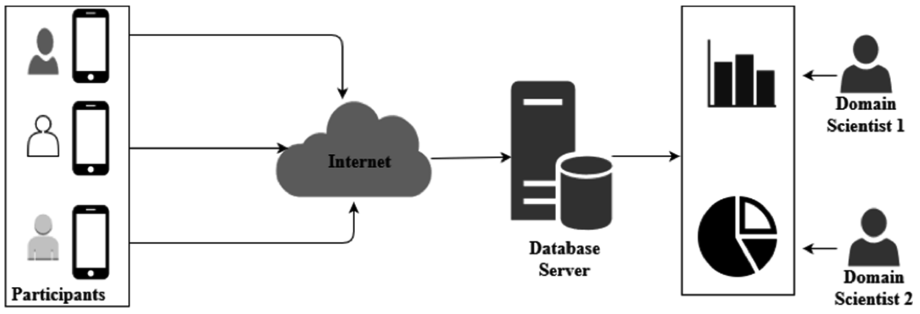

Fig. 1. Conventional model for participatory sensing applications

Domain Scientists. Domain scientists subscribe to the service and access the data gathered by participants. They are experts who use the data collected by participants to analyze the target phenomenon.

Participatory Sensing Infrastructure Provider. Application infrastructure providers are initiators of participatory sensing campaigns. They are responsible for designing, implementing, deploying, and managing the applications.

3 System Architecture

We developed a web-based platform for domain scientists such as social researchers, ecologists, and environmentalists. Using this platform domain scientists can specify AOIs on a map (i.e., the areas from which they are interested in collecting data). Along with AOIs, stop limit (i.e., the number of data entries required for each targeted area) can be specified to prevent additional data collection from a particular area. As depicted in Fig. 2, marked AOI along with stop limits are stored in a database server. When participants collect data using the mobile application, the nearby AOIs are shown on the map as a guide to the participants.

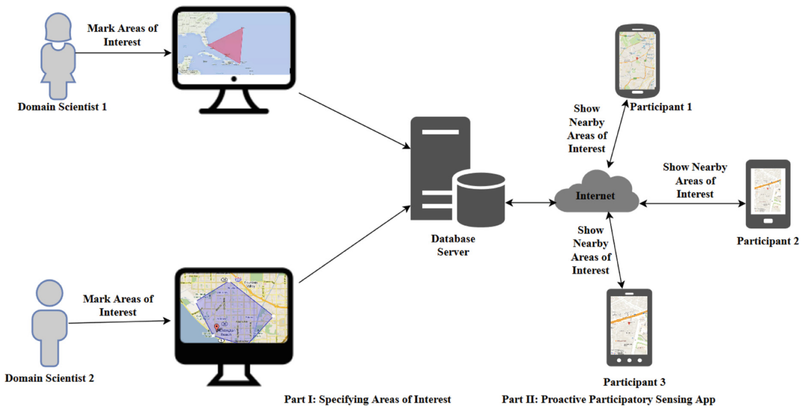
Mark Areas of Interest
Domain Scientist 1
Mark Areas of Interest
Domain Scientist 2
Database Server
Internet
Show Nearby Areas of Interest
Participant 1
Show Nearby Areas of Interest
Participant 2
Show Nearby Areas of Interest
Participant 3
Part I: Specifying Areas of Interest
Part II: Proactive Participatory Sensing App

Fig. 2. System architecture

This system is divided into two parts. The first part is designed to identify AOIs by domain scientists, and the second part is designed to obtain data from targeted areas. To identify the AOIs, a web portal is implemented where domain scientists provide the coordinate of their targeted areas upon authorization. They can search for any region on the map and mark the areas, in the form of rectangles as depicted in Fig. 3. Identification of AOIs also allows domain scientists to specify the number of required entries for that area as the stop limit. This information is passed to the server to be stored in a spatial database. PostgreSQL along with PostGIS as a spatial database extender is used to store the spatial and non-spatial data.

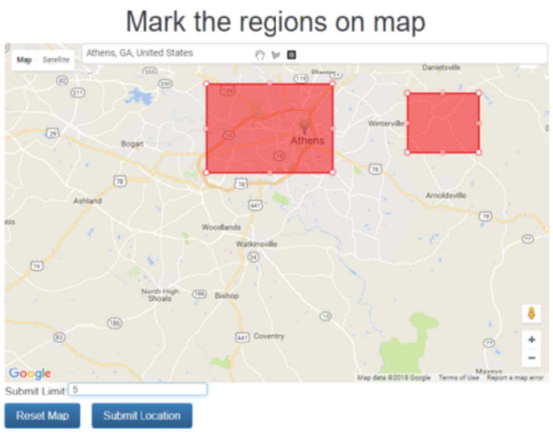

Fig. 3. Identifying Areas of Interest (AOIs)

Proactive participatory sensing approach obtains data from the database server and enables targeted and proactive participatory sensing by showing nearby AOIs to participants. As shown in Fig. 4, this system consists of three modules: data collection platform for mobile applications on the left, web service for AOIs retrieval, and a database server.

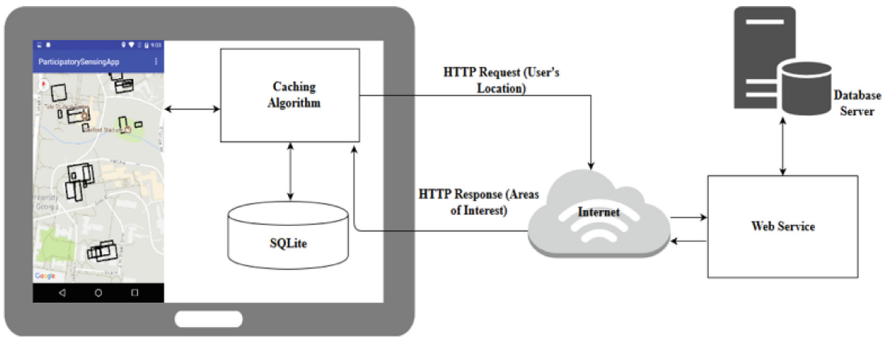

Fig. 4. Proactive participatory sensing

Our data collection platform for mobile application is developed using Android SDK. As depicted in Fig. 5, the app allows participants to specify the radius (from now on, we call it "specified distance") and then shows the AOIs within that radius to the users. Considering that the participants are constantly moving, repeatedly updating the AOIs requires continuous connection and querying from the database which leads to performance degradation. To overcome this challenge, we prefetch targeted areas on the clients' device and present a caching algorithm to improve the performance.

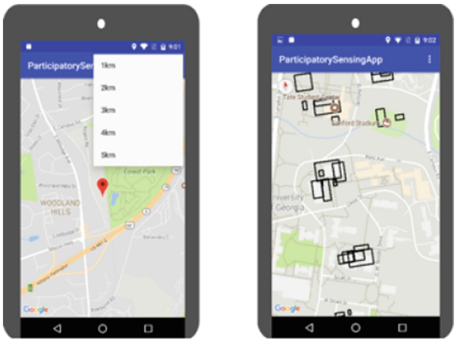

Fig. 5. Android application showing nearby AOIs

The caching algorithm is illustrated in Fig. 6. The user's locations are constantly tracked in the background. Users specify the distance within which they are willing to see the available AOIs (it is called "specified distance"). In addition, we use the term "extra-prefetched distance" to refer to the extra amount of data which the application retrieves from the database server and prefetches to the cache to have a better performance. In other words, the application retrieves more data than required from the database so as to perform more efficiently while the users are moving. In the next step, the whole data, i.e., all the AOIs within a circle that its radius is equal to the specified distance plus the extra-prefetched distance, are inserted into the cache. For example, if a participant specifies to see all the AOIs within 1 km radius and the application is set to 2 km of extra prefetched distance, the AOIs which are within 3 km distance from the current location of the participant will be retrieved from the database server and inserted into the mobile phone's cache. Initially, only the AOIs which are within 1 km radius will be shown to the participant. As they move, the application keeps fetching data from the cache and shows them to the user. The algorithm also keeps calculating the distance between the initial location and the current location of the participant to make sure they are still within the valid scope. The valid scope is defined as the area where its data is available in the cache. For instance, in the last example, if the participant walks for more than 2 km, they go beyond the valid scope, and the data in the cache needs to get updated. In that case, the application retrieves the updated data from the database server, clears the cache, insert the updated data into the cache, and again keeps fetching from the mobile's cache.

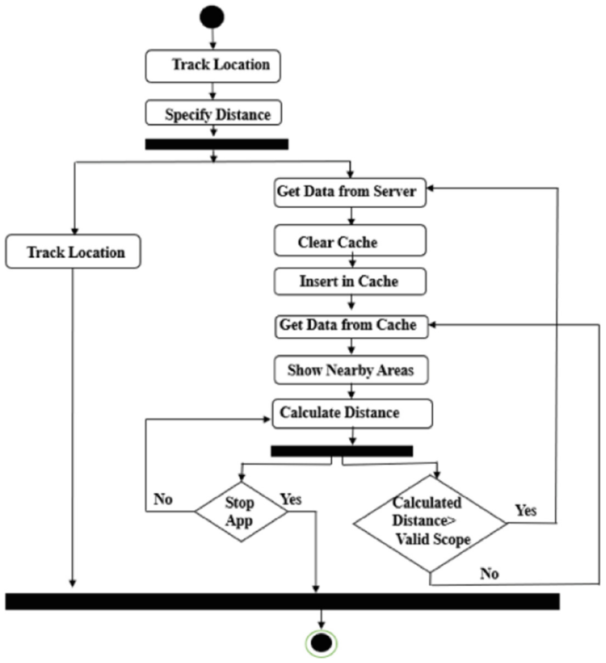

Fig. 6. Basic caching algorithm

For caching targeted areas on the client side, we used SQLite on mobile phones as an embedded relational database. It is serverless, highly portable, easy to use, efficient, and reliable. SQLite is used in numerous applications such as in Apple's Aperture photography software and the Safari web browser. We also used SpatiaLite as a spatial extender of SQLite for caching AOIs (i.e., rectangles) that are retrieved from the database server.

4 Experimental Study

In this section, we discuss the performance of our system for varying extra prefetched distances. We show the performance of our algorithm on three smartphones with different configurations for different caching scenarios.

Experimental Setup. Three mobile phones used for our analyses are OnePlus 3T with 6 GB RAM and 64 GB of internal memory, Samsung Galaxy Note 4 with 4 GB RAM and 32 GB of internal memory, and Nexus-5 with 2 GB RAM and 16 GB of internal memory. We tested the results of our algorithm on a backend server using PostgreSQL v9.6 with spatial extender PostGIS v2.4, and MongoDB v3.6. Our database resides on a Windows server with 8 GB memory using Intel Core i5 2.7 GHz processor. It should be mentioned that we used the geometry data type in SQLite to store each AOI which is 120 bytes. Considering that the primary key associated with each AOI is of type integer (2 bytes), therefore, each row in the cache takes 122 bytes.

Our dataset consists of one million rectangles (i.e., domain scientists' AOIs). We randomly generated twenty thousand rectangles in each state of the United States. Each state is considered as a bounding box, and the random points are generated within each box. Then, we generated AOIs by randomly choosing two points which are the two ends of each rectangle's diagonal. Generated rectangles could be either overlapping or non-overlapping. On average, there are 46 AOIs in each 1 km radius.

Experimental Details. In order to test our algorithm, we simulated the scenarios in which a participant is moving for five kilometers from the initial location while collecting data. We assume that the participant is only interested to see the AOIs within 1 km of their location at any given point of time. In order to compare the performance of different caching approaches, we assumed that the number of AOIs updates every 0.5 km (therefore, the location information is captured every 0.5 km). We created an array of locations for each route and calculated the average time required to show AOIs if the user moves for 5 km by varying extra prefetched distances. For instance, if the extra prefetched distance is set to 2 km, the application retrieves the AOIs within 3 km radius from the database and inserts into the cache. At the start location, the response time is equal to the time required for retrieving data from the database, clearing the cache, inserting into the cache, fetching the AOIs within 1 km from the cache, and showing them to the user. At locations 0.5 km, 1 km, 1.5 km, and 2 km, the response time equals to the fetching time from the cache. Going one step further, at 2.5 km, some of the AOIs within 1 km radius is not in the cache anymore. Therefore, the application needs to retrieve the new data from the database server again and follow the steps depicted in Fig. 6. We continue the same steps up to the 5th kilometer of the route and average the response time of all 11 points (start point, end point, and the 9 points in the middle). We assign this average response time to that route. We compared the results of our algorithm by varying the extra prefetched distances for which the data were cached. Table 1 shows the percentage of cache misses associated with different prefetched distances for the routes used for the following experiments.

We perform the following three experiments to test the caching on different mobile phones on LTE network.

Table 1. Prefetched distance and cache misses.

Prefetched distance (km)	Percentage of cache misses
0	100%
1	36%
2	27%
3	18%
4	18%

4.1 First Approach: Basic Caching

In the first experiment, we test the performance of the caching algorithm exactly as explained in Fig. 6. In other words, this experiment includes the following steps:

1. Retrieving the data (both the specified and extra- prefetched distance) from the database.
2. Clearing the cache.
3. Inserting into the cache.
4. Fetching the required data from the cache.
5. Showing the results for the specified distance to the user.

In this experiment, the x-axis represents the extra prefetched distances, and the y-axis shows the average response time to show the AOIs. As shown in Fig. 7, we observed that the performance of the algorithm improves as the extra prefetched distance increases up to a certain point, then it degrades for 4 km of extra prefetched data. We can find the reason in Table 1, which shows the percentage of the cache misses for 3 km and 4 km extra prefetched distance is the same. Therefore, inserting more data into the cache and fetching from it takes more time. In Fig. 7, the depicted values for the extra prefetched distance of zero shows the time required to retrieve the data from the database server (at that step, the cache is still empty).

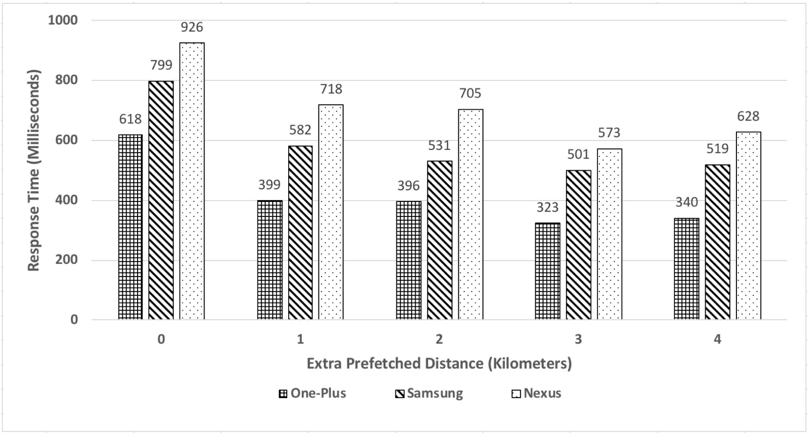

Fig. 7. Performance of the basic caching

To have a better understanding of the caching performance, we calculated the cache insertion and fetching time on different phones. Figure 8 depicts the time it took to insert the AOIs into the cache of different mobile phones. There is a considerable increase in the insertion time as the number of AOIs increases. Figure 9 depicts the fetching time for the same number of AOIs. The difference between the insertion and fetching time is noticeable. On average, the insertion time is 7.5 times more than the fetching time. Therefore, we got to know that cache insertion is a very expensive operation and to improve the total response time in such applications, we need to mainly focus on the insertion operation.

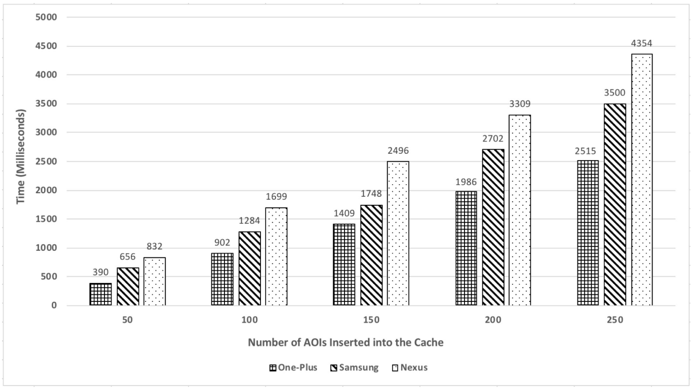

Fig. 8. Inserting data into the cache

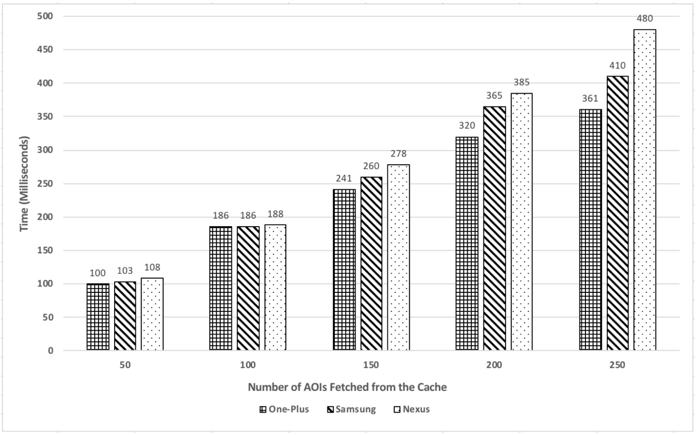

Fig. 9. Fetching data from the cache

4.2 Second Approach: Caching in the Background with Single Database Call

In the second experiment, we test the performance of the algorithm by inserting the data into the cache in the background. In other words, this experiment includes the following steps:

1. Retrieving data (both the specified and extra-prefetched distance) from the database. This step consists of two queries in one database hit, i.e., one query for the whole data and the other one for the specified distance only.
2. Showing the results for the specified distance to the user.

3. Clearing the cache in the background.
4. Inserting into the cache in the background (insertion is an expensive operation).
5. Fetching the required data from the cache for the following locations of the user.

Instead of inserting the data into the cache (which is an expensive operation) and then showing it to the user, in this experiment we first retrieve both extra prefetched and specified distance AOIs from the database server. Then, the application shows the specified distance AOIs directly to the user, and the extra prefetched distance AOIs are inserted into the cache in the background as participants are moving. As shown in Fig. 10, there is a drastic increase in the performance of this approach compared to the first approach. In this figure, when the extra prefetched distance in zero, it means that the application retrieves the data from the database server and nothing is inserted into the cache yet.

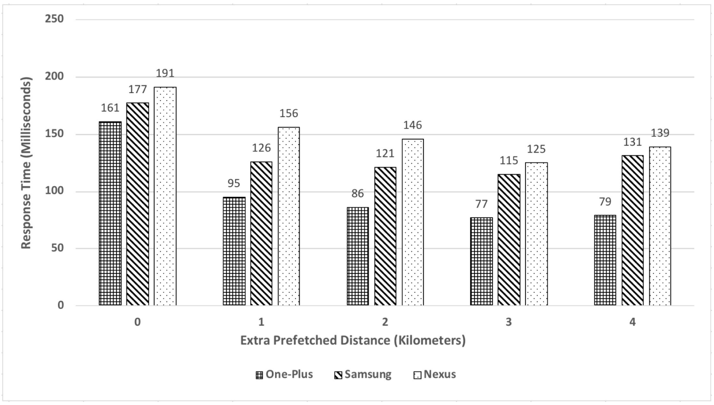

Fig. 10. Performance of caching in the background with single database call

4.3 Third Approach: Caching in the Background with Two Database Calls

In the third experiment, we test the performance of the algorithm same as previous experiment, and the only difference was the fact that instead of retrieving the whole data in the first step, we only retrieved AOIs within the specified distance to be shown to the users. In the next step, we query the database for the second time to retrieve the whole data. In other words, this experiment includes the following steps:

1. Retrieving data (specified distance only) from the database.
2. Showing the results for the specified distance to the user.

3. Retrieving data (both the specified and extra-prefetched distance) from the database in the background.
4. Clearing the cache in the background.
5. Inserting into the cache in the background.
6. Fetching the required data from the cache for the following locations of the user.

In this approach, while the application is showing the AOIs to the users, retrieving the whole data and the other steps happen in the background. As depicted in Fig. 11, we can see a performance increase on all the phones. In this figure again, the extra prefetched distance of zero shows that there is no data in the cache and the depicted numbers show the time it took to retrieve the data from the database server.

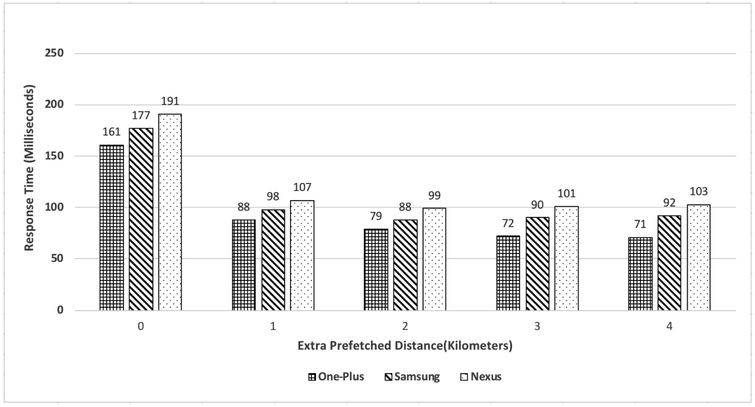

Fig. 11. Performance of caching in the background with two database calls

Comparing the Three Approaches. In order to make an unbiased conclusion, we repeated the experiments associated with each of the three caching approaches on 10 different routes. Figure 12 depicts the average response time to show the AOIs to the users. We observed that the third approach performs 5.48 times faster than the first approach and 10% faster than the second approach. Therefore, for proactive and targeted participatory sensing applications, the third caching approach would be the best choice. Although it is reasonable to have a limit for caching the data, we cannot generalize the observation in our experiments which shows going beyond 3 km of extra prefetched distance leads to performance degradation. In fact, the optimum cache limit is application- specific, and it pretty much depends on the number of nearby AOIs.

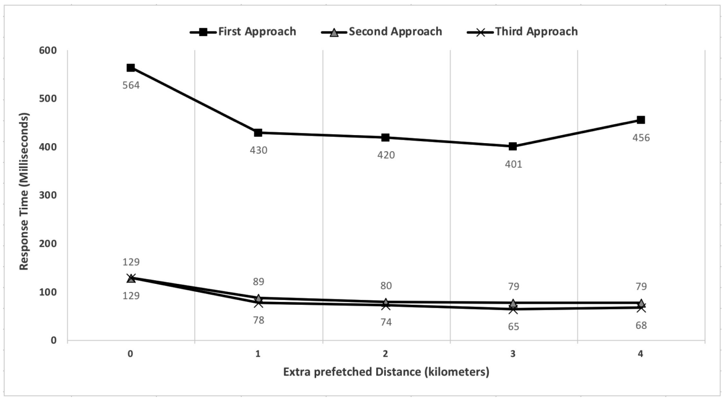

Fig. 12. Performance comparison of the three approaches

5 Related Works

In this section, we review some related studies. Caching the data has been widely used in mobile environments for improving access time. However, cached data becomes obsolete due to the movement of users known as location-dependent data invalidation [20]. Several approaches are proposed to overcome this challenge.

Zheng et al. [19] proposed location dependent data cache invalidation and replacement under a geometric model. They introduced polygon endpoints and app circle invalidation schemes for representing the valid scopes (i.e., regions within which data values are valid). Polygon endpoint records coordinates of polygons representing the valid scope. However, when there are a large number of endpoints, they will consume a substantial portion of the cache space. They also introduced an alternative approximation scheme, which uses inscribed circles to approximate a polygon. However, this scheme treats valid data as invalid for data points which are outside the circle but within the polygon. In order to make a balance between the overhead and precision, a caching-efficiency-based method was proposed. Various cache replacement policies such as probability area and probability area inverse distance in location-dependent services were also proposed in this paper. Access probability, data distance, and valid scope were three factors considered for cache replacement. In the probability area, valid scope and access probability were considered for calculating the cost function, whereas, in the probability area inverse distance, all three factors, i.e., access probability, valid scope, and data distance were considered for calculating the cost function. Both policies choose data with least cost as the victim.

Xu et al. [22] addressed the issue of location-dependent cache invalidation under a cell-based systems location model. Bit vector with compression grouped vector with compression, and implicit scope information is the three methods that were proposed. In this study, it is assumed that each geographical area is partitioned into service areas and each service area can cover one or multiple cells. Also, each service area is associated with an ID for identification purposes which is broadcasted periodically to all mobile clients in that service area. Bit Vector with compression uses bit vector, and its length is equal to the service areas in a system. Every cache item is associated with a bit vector; however, associating every cache item with bit vector is an overhead for a large system. Grouped vector with compression was proposed as a solution to bit vector with compression, where the whole geographical area is divided into disjoint districts and all the data service areas within a district form a group. In Implicit Scope Information model, the database is divided into multiple logical sections based on a valid scope. Data in the same logical section has the same location validation information.

6 Future Works and Improvements

MongoDB is one of the most popular open-source NoSQL databases. For the sake of performance improvement, we analyzed the third caching approach using MongoDB to compare with PostgreSQL. We created the same dataset for MongoDB and performed the same experiments. As depicted in Fig. 13, the performance is consistently better on MongoDB as opposed to PostgreSQL. Therefore, for future work, we are migrating to MongoDB.

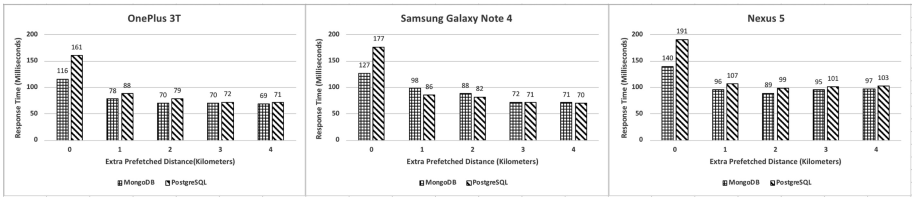

Fig. 13. Performance of the third approach on different database servers

We would also like to compare the performance of the discussed approaches by updating the cache rather than clearing the cache and inserting data into it again. Considering that in participatory sensing applications users often take a loop-like trajectory where they end up near their start location, there are some overlaps with the previously visited AOIs. As a result, updating the cache may outperform our current approaches.

7 Conclusion

Widespread use of mobile phones with embedded sensors has introduced the term participatory sensing. Despite the many advantages of such frameworks, there are some challenges and limitations. Most of the participatory sensing applications are passive, and there is little or no communication between the domain scientists and the participants. In this study, we present a mobile and web-based approach to enable domain scientists to acquire crowd-sensed data from particular areas of interest adequately. Domain scientists mark the areas and participants are proactively informed about the sensing opportunities near their current location. In order to show the nearby AOIs to the participants, data need to be retrieved from a database server frequently, which leads to performance degradation. In order to increase the performance, we introduce a caching approach that stores and prefetches the nearby AOIs locally from the participant's mobile phones. We analyze the performance of the algorithm for three different caching approaches on different mobile phones and present the approach with the best performance.

Acknowledgment. This research has been partially funded by the National Science Foundation (NSF) under grants CCF-1442672 and SCC-1637277 and gifts from Accenture Research Labs. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or other funding agencies and companies mentioned above.

References

1. Christin, D., Reinhardt, A., Kanhere, S.S., Hollick, M.: A survey on privacy in mobile participatory sensing applications. J. Syst. Softw. **84**(11), 1928–1946 (2011)
2. Shilton, K., Estrin, D.: Participatory sensing and new challenges to US privacy policy
3. Guo, B., Yu, Z., Zhou, X., Zhang, D.: From participatory sensing to mobile crowd sensing. In: 2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS), pp. 593–598. IEEE (2014)
4. Tonekaboni, N.H., Kulkarni, S., Ramaswamy, L.: Edge-based anomalous sensor placement detection for participatory sensing of urban heat islands. In: 2018 IEEE International Smart Cities Conference (ISC2), pp. 1–8. IEEE (2018)
5. Ganti, R.K., Ye, F., Lei, H.: Mobile crowdsensing: current state and future challenges. IEEE Commun. Mag. **49**(11), 32–39 (2011)
6. Kapadia, A., Kotz, D., Triandopoulos, N.: Opportunistic sensing: security challenges for the new paradigm. In: 2009 First International Communication Systems and Networks and Workshops, pp. 1–10. IEEE (2009)
7. Kanhere, S.S.: Participatory sensing: crowdsourcing data from mobile smartphones in urban spaces. In: Hota, C., Srimani, Pradip K. (eds.) ICDCIT 2013. LNCS, vol. 7753, pp. 19–26. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36071-8_2
8. Mathur, S., et al.: Parknet: drive-by sensing of road-side parking statistics. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, pp. 123–136. ACM (2010)
9. Mohan, P., Padmanabhan, V.N., Ramjee, R.: Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, pp. 323–336. ACM (2008)
10. Deng, L., Cox, L.P.: Livecompare: grocery bargain hunting through participatory sensing. In: Proceedings of the 10th Workshop on Mobile Computing Systems and Applications, p. 4. ACM (2009)
11. Youdale, N.: Haze watch: database server and mobile applications for measuring and evaluating air pollution exposure. Electrical Engineering and Telecommunication School, University of New South Wales, Sydney, NSW, Australia, Technical report (2010)
12. Von Kaenel, M., Sommer, P., Wattenhofer, R.: Ikarus: large-scale participatory sensing at high altitudes. In: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, pp. 63–68. ACM (2011)

13. Maisonneuve, N., Stevens, M., Niessen, M.E., Steels, L.: NoiseTube: measuring and mapping noise pollution with mobile phones. In: Athanasiadis, I.N., Rizzoli, A.E., Mitkas, P. A., Gómez, J.M. (eds.) Information Technologies in Environmental Engineering. Environmental Science and Engineering. Springer, Berlin (2009). https://doi.org/10.1007/978-3-540-88351-7_16
14. Kanjo, E.: NoiseSPY: a real-time mobile phone platform for urban noise monitoring and mapping. Mob. Netw. Appl. **15**(4), 562–574 (2010)
15. Kim, S., Robson, C., Zimmerman, T., Pierce, J., Haber, E.M.: Creek watch: pairing usefulness and usability for successful citizen science. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2125–2134. ACM (2011)
16. Ganti, R.K., Pham, N., Ahmadi, H., Nangia, S., Abdelzaher, T.F.: GreenGPS: a participatory sensing fuel-efficient maps application. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, pp. 151–164. ACM (2010)
17. Reddy, S., Parker, A., Hyman, J., Burke, J., Estrin, D., Hansen, M.: Image browsing, processing, and clustering for participatory sensing: lessons from a DietSense prototype. In: Proceedings of the 4th Workshop on Embedded Networked Sensors, pp. 13–17. ACM (2007)
18. Eisenman, S.B., Miluzzo, E., Lane, N.D., Peterson, R.A., Ahn, G.-S., Campbell, A.T.: BikeNet: a mobile sensing system for cyclist experience mapping. ACM Trans. Sens. Netw. (TOSN) **6**(1), 6 (2009)
19. Zheng, B., Xu, J., Lee, D.L.: Cache invalidation and replacement strategies for location-dependent data in mobile environments. IEEE Trans. Comput. **51**(10), 1141–1153 (2002)
20. Ren, Q., Dunham, M.H.: Using semantic caching to manage location dependent data in mobile computing. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, pp. 210–221. ACM (2000)
21. Kanjo, E., Bacon, J., Roberts, D., Landshoff, P.: MobSens: making smart phones smarter. IEEE Pervasive Comput. **8**(4), 50–57 (2009)
22. Xu, J., Tang, X., Lee, D.L., Hu, Q.: Cache coherency in location-dependent information services for mobile environment. In: Leong, H.V., Lee, W.-C., Li, B., Yin, L. (eds.) MDA 1999. LNCS, vol. 1748, pp. 182–193. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-46669-X_16